Comparison of Neural Architectures for Sentiment Analysis of Movie Reviews

Sören Hübner*, Nicolás Pérez de Olaguer*
{8huebner, 7perez}@informatik.uni-hamburg.de

*Department of Informatics, University of Hamburg
Knowledge Processing in Intelligent Systems: Practical Seminar
Knowledge Technology, WTM

***Abstract*—Infering sentiments from text is not always a straightforward procedure. Language systems are usually complex and the use of figures of speech such as irony may lead to wrong conclusions. Several machine learning techniques can be helpful to unveil the underlying complexity of such texts, although prediction accuracy is always an important issue. In this paper, we use the well-known Internet Movie Database (IMDb) data set [1] and try to predict the sentiment behind a movie review. We compare the performance and applicability of several machine learning approaches including artificial neural networks. This work derives from a Kaggle [2] competition held in 2016 and is based around our implementation of several different approaches. We used the neural network library keras [3] for the python programming language [4] to achieve acceptable results with three distinct neural network architectures.**

I. Introduction

There are several tasks where humans are still better than machines. More specifically, when we talk about human communication and sentiment analysis, algorithms are prone to fail. Since the understanding of communication is not unique and can have several interpretations depending on the sender and the receiver, it can be beneficial to explore how automatic techniques can deal with it. Moreover, the inherent complexity of language makes this even harder. For example, literary nuances such as irony or metaphors increase the difficulty of the classification task.

The goal of this paper is to classify movie reviews into two distinct categories: 'positive' and 'negative'. We use the text of the review, embed it in a certain way and feed the resulting vector into one of three different neural network architectures.

The sentiment mining of expressions or texts is useful for research studies or marketing purposes. In table I, we show an example of two different movie reviews and their corresponding sentiment from the used corpus [1]. A feature of the IMDb movie reviews is that not all reviews come from cinema experts but from all users of the internet platform. Therefore, reducing manual effort to extract sentiment is useful for the movie industry to know whether a film is well accepted by the general audience based on the reviews. In related work [5], the authors perform sentiment analysis of Tweets with an important distinction between positive, negative and objective statements. Nevertheless, in our case, we do not have the third 'objective' class, because the dataset only includes 'positive' reviews with a given score of 7 or higher (on a scale from 1 to 10) and 'negative' ones with a score of 3 or less.

II. Related Work

First approaches to sentiment and mood mining were relying on basic assumptions. In [6] they do a comprehensive review of the more useful techniques for information retrieval. In [7] they infer mood from web blogs using SVM (support vector machine) and CRF (conditional random field) techniques. It is interesting how taking the assumption of relying on the last sentence of the post gives better accuracy for sentiment prediction.

Since relevance of web blogs has decayed, in [5], they do sentiment analysis for tweets. Again, it is different from our scenario since microblogging can take only up to 140 characters and in our case, we have freedom from the reviewer. However, they reach around 80% accuracy using a bi-gram model and a Naive Bayes classifier. More novel approaches can be found in [8] where neural architectures are applied to the same twitter task with an accuracy of 86,4% using Stanford Twitter Sentiment from Stanford University. It is also worth to mention that the developed tasks proceed from a Kaggle competition [2]. In that challenge, the top accuracy scores get to 97% of accuracy.

III. Background

Artificial neural networks use linear algebra to 'learn' things, approximate functions, classify data points and do many more things. It consists of several 'neurons' which are grouped into layers which themselves are connected with each other. Those connections are weighted to simulate stronger or softer bonds. Some data vector is given as an input and the network then produces an output vector. This type of architecture has the roots in the brain. The artificial neuron is a mathematical entity that has some weights and an activation function.

Review	Sentiment
"I dont know why people think this is such a bad movie. Its got a pretty good plot, some good action, and the change of location for Harry does not hurt either. Sure some of its offensive and gratuitous but this is not the only movie like that. Eastwood is in good form as Dirty Harry, and I liked Pat Hingle in this movie as the small town cop. If you liked DIRTY HARRY, then you should see this one, its a lot better than THE DEAD POOL. 4/5"	Positive
"I watched this video at a friend's house. I'm glad I did not waste money buying this one. The video cover has a scene from the 1975 movie Capricorn One. The movie starts out with several clips of rocket blow-ups, most not related to manned flight. Sibrel's smoking gun is a short video clip of the astronauts preparing a video broadcast. He edits in his own voice-over instead of letting us listen to what the crew had to say. The video curiously ends with a showing of the Zapruder film. His claims about radiation, shielding, star photography, and others lead me to believe is he extremely ignorant or has some sort of ax to grind against NASA, the astronauts, or American in general. His science is bad, and so is this video."	Negative

Table I: Sample of one negative and one positive review from the IMDb dataset.

Neuron cells receive connections from other cells and fire when they receive electrical impulses from other connected neurons. It is know, that the more a connection is used, the more stronger it will get.

A loss or **cost function** is used to turn the learning aspect of neural networks into an optimization problem. It associates some loss or cost with wrong outputs and 'punishes' a network for performing in an undesirable manner. Usually good answers are reinforced and bad ones are discouraged on the basis of a loss function and a process called 'back-propagation', where the neurons which are more responsible for a certain outcome are encouraged or discouraged the most. Nowadays, artificial neural networks, (ANNs) have been proven to solve a lot of classification and regression problems.

In the beginning of our experiments we used **word2vec** [9], a special embedding technique for words. Word2vec groups word vectors based on learned semantics. Although word2vec is not trained with any specific language grammar, it is able to make similarities among two languages. Later, we switched to another, similar embedding, see section IV for further details.

We used keras [3] for our implementation in python. Keras is a popular neuronal network framework based on Tensorflow [10].

IV. Proposed Approaches

For our analysis, we implemented three different approaches to solve the problem of classification. Thus, we compare three different solutions all by using neural architectures. We used Python with Keras backend to train our models. Consisting of a Multi-Layer Perceptron (MLP), a Convolutional Neural Network (CNN) and a long short-time memory (LSTM).

To load the data we use the already built-in Keras function to import the IMDb corpus. The reviews have been already preprocessed and words are indexed to its number of occurrence in the corpus. We set the maximum of words up to 5000 and a maximum review length to 500 words.

For the embedding vector, we also use a built-in Keras function to train it. We specify an output length of 32, resulting in a $500 * 32$ embedding vector. The default Keras function works similarly to word2vec. The difference is that with the Keras embedding layer we try to minimize the loss function. On the other hand, word2vec gives an embedding vector that keeps the semantic relation between words. Consequently, the embedding Keras vector might be useful for capturing only polarity of emotions. Nevertheless, the idea still keeps similar for both approaches.

A. Embedding

As mentioned in section III, we used to use the word2vec embedding for our experiments. We implemented a version of it ourselves, trained it and got good results. We later learned about the embedding layer [11] of the keras library which transforms all input into a special vector. This layer uses a similar technique as word2vec, see [12] for a comparison of the two methods. For us, the embedding layer delivered almost the same performance while being easier to use and slightly faster.

B. Multi Layer Perceptron

MLPs [13], [14] consist of one input layer, one output layer and one or more densely connected layers inbetween. It is one of the most simple existing artificial neural networks. Nonetheless, it is known that works as good as other more complex architectures with simple tasks. MLPs were first designed to make class prediction of a certain data. They need labelled training data to adjust their weights and find decision boundaries between data. It is the most simpler, but powerful, ANN used. Since we are dealing with a binary classification problem, it is a good candidate for having good results using less computational resources than other approaches. In our scenario, we have created an MLP with 250 fully connected neurons with a rectified linear unit (ReLU) activation function. Detailed visualization of the structure of the model can be seen in Figure 1.

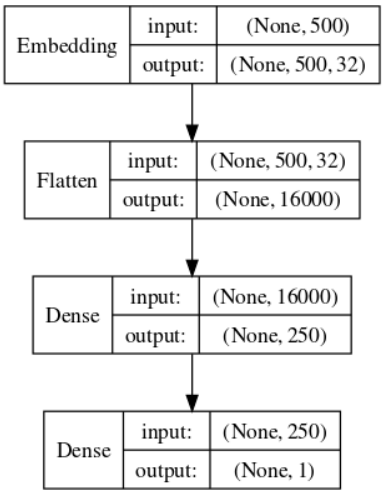

Figure 1: A detailed structure of the MLP. The embedded vector gets flattened before being send to a fully connected layer. We use binary classification.

C. Convolutional Neural Network

Convolutional neural networks (CNNs) [15], [16] are neural architectures that can learn features from data vectors themselves. They use specialised layers to learn and apply filters to the dataset. CNNs have been proven to overcome many other neural architectures in terms of accuracy, specifically in the vision research field. The typical CNN architecture consists of a convolution layer, where a linear convolution is applied to the input, with a certain kernel size. A max-pooling layer is added to the architecture to reduce the size of the convolution. Moreover, a fully connected layer is appended to get a classification outcome. The convolution and max-pooling layers can be added arbitrarily to the network to get to a certain vector size. It is known that for every convolutional layer it is added, different level of features can be learned. First layers of the network can learn low levels of features and the more you add, higher complexity features. That is why the 'Deep Learning' term is used, to describe a certain network with more than a couple of layers. The shortcoming of deep networks is that the number of weights to tune while training is huge. Therefore, is computationally expensive to train such networks. In our case, we only add one time the convolutional layer. We expect to learn basic features about the data. Therefore, in Figure 2 we can see how our architecture looks like. We use 32 filters and a $3 * 3$ kernel size for the convolution layer. The activation function is again ReLU. For the max-pooling layer, we reduce our size by a factor of 2. Then we add a fully connected layer of 250 neurons along with the last binary classification.

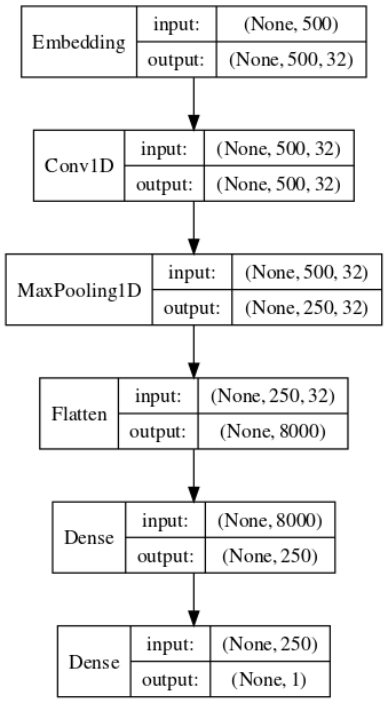

Figure 2: The CNN model. The special features are the convolutional layer and the sub-sampling layer that reduces our feature space. The subsequent part is equal to the MLP.

D. Long Short-Time Memory Network

LSTM networks [17] are a type of recurrent neural network (RNN) which are used a lot in the natural language processing field and other fields working with time dependant data. They can learn more long-term dependencies than a simple RNN. Since we are working with texts that are longer than a sentence, LSTM is also a good candidate. In Figure 3 we can see a detailed explanation of the architecture used. We use $n = 100$ as the dimensionality of the output space.

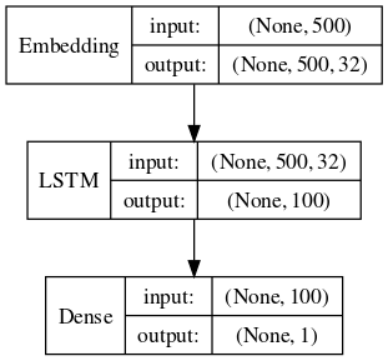

Figure 3: Structure of the used LSTM architecture. In keras exists a LSTM layer which abstracts from the common gates and structures of a LSTM for easier implementation.

E. Short Discussion of the used Architectures

In each of the above models, we need to add a last layer into the network. The output layer is responsible for classifying between negative and positive reviews and thus consists of only one neuron. The loss function of our models is the binary cross entropy and we are especially interested in the accuracy of the trained model.

We used the three introduced architectures because of their popularity and their diverse properties. All three belong to the most used types of neural networks [18]. The MLP is supposed to give a baseline for the reults of the other networks. With the CNN, we tried to overcome the literal nuances problem. We believed that by using this self-learning feature approach such complexities could be overcame. By using hte LSTM cells, we tried to leverage the fact that language is time dependent. It is clear that a word has several relation to its precedent and future words. With the three different architectures we have benefitts and drawbacks for using them, therefore is worth try them three. Moreover, all networks achieved acceptable results.

V. Results

We achieved similar results with all three different neural networks. The MLP and CNN both took between ten and fifteen minutes to train on a single CPU and the LSTM over two hours. Given the absence of big differences in the prediction accuracy, training the LSTM is not really worth it. Our best performing network is the presented CNN structure, trained for just one epoch, with a prediction accuracy of 87.80% on the test set. In the Kaggle challenge [2] would have placed 267th out of the 578 participants which is above average and better than the three baseline classifiers [19] for this competition. In table II, we can find a described numeric values for the results for the presented architectures.

	MLP	**CNN**	**LSTM**
1 epoch	87.33	87.80	84.67
10 epochs	86.16	87.59	86.29

Table II: Accuracy on the test set of the different architectures after one and after ten epochs.

Detailed plots of the train and test loss and accuracy can be found in the appendix. It is clear that after the first training epoch, the test accuracy stays about the same, just the training accuracy increases. Training for more than one epoch does not bring any merit. We can spot some overfitting because the test loss increases as the test accuracy stays the same. The network makes wrong decisions while being more sure about it. This could mean that the networks are bigger than needed for the task. Dropout could be used to circumvent this as well as a smaller batch size and smaller learning rate.

VI. Conclusion

On the given dataset, acceptable results can be achieved with a variety of different methods. Capturing time-related dependencies (as a LSTM does) or learning filters (as a CNN does) does not seem to add a lot of performance for this specific use case. But it could also be either the task, the dataset or even the preprocessing and embedding that diminishes the variety of our results.

Our experiements show that for certain machine learning tasks, there can be several different approaches which despite structural differences manage to perform similar. In such a case, it is advisable in regard of overall efficiency to choose a network that is easier or faster to train, like the MLP or CNN in our experiments. It is plausible that further optimizations and tweaking of the hyperparamters would lead to more apparent differences between the networks and other performance characteristics.

References

[1] Imbd data files. https://datasets.imdbws.com/.

[2] Bag of words meets bags of popcorn. *Kaggle Inc.*, 2015.

[3] Keras: The python deep learning library. https://keras.io/.

[4] Python software foundation. https://www.python.org/.

[5] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. 2010.

[6] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2008.

[7] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271, 2004.

[8] Cicero dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[10] Tensorflow: An open source machine learning framework for everyone. https://www.tensorflow.org.

[11] Stackexchange: How the embedding layer is trained in keras embedding layer. https://stats.stackexchange.com/questions/324992/how-the-embedding-layer-is-trained-in-keras-embedding-layer.

[12] Daniel López-Sánchez, Jorge Revuelta Herrero, Angélica González Arrieta, and Juan M. Corchado. Hybridizing metric learning and case-based reasoning for adaptable clickbait detection. *Applied Intelligence*, 48(9):2967–2982, Sep 2018.

[13] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[14] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[15] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[16] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[19] Kaggle: Imdb competition baselines. https://www.kaggle.com/c/word2vec-nlp-tutorial#description.

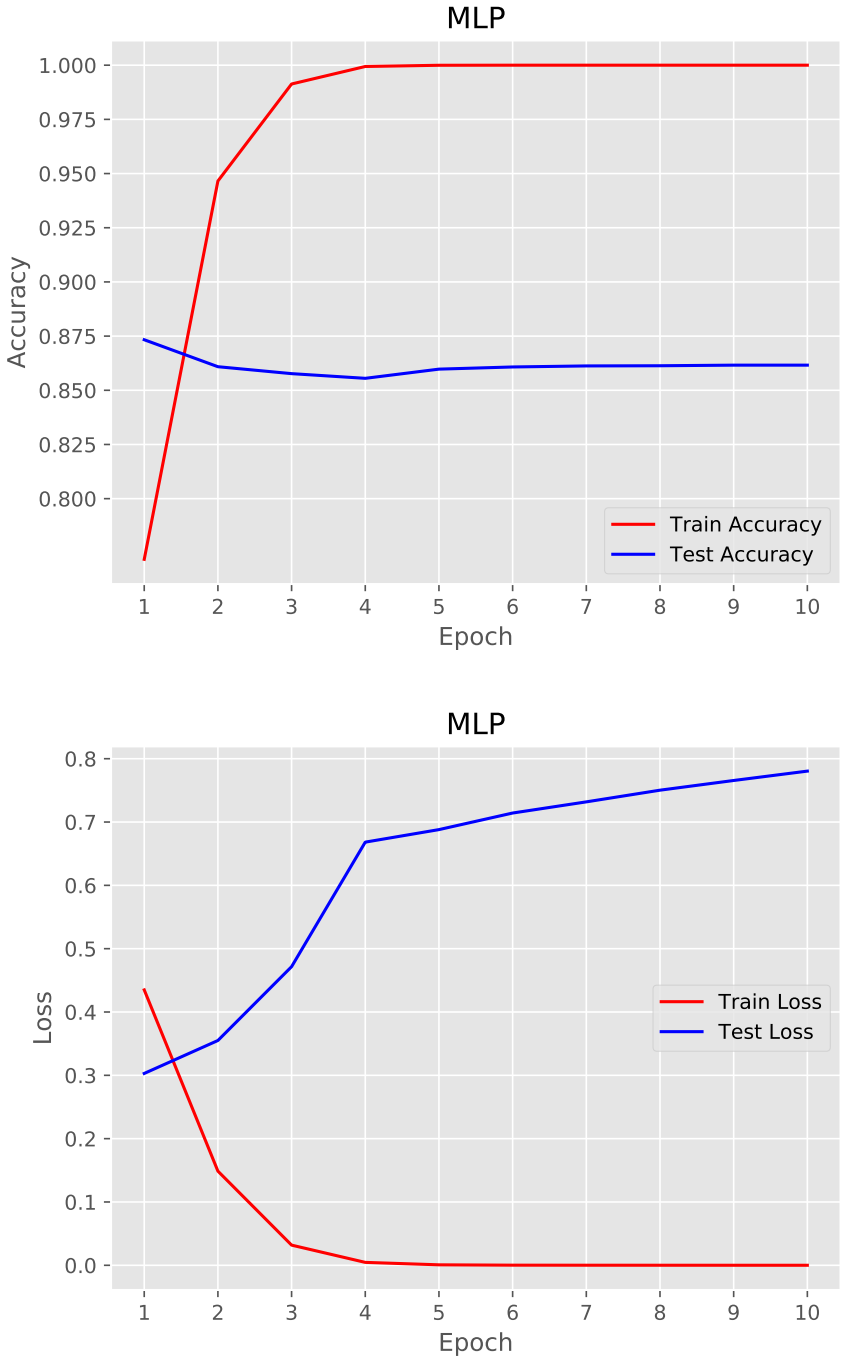

Figure 4: MLP accuracy and loss.

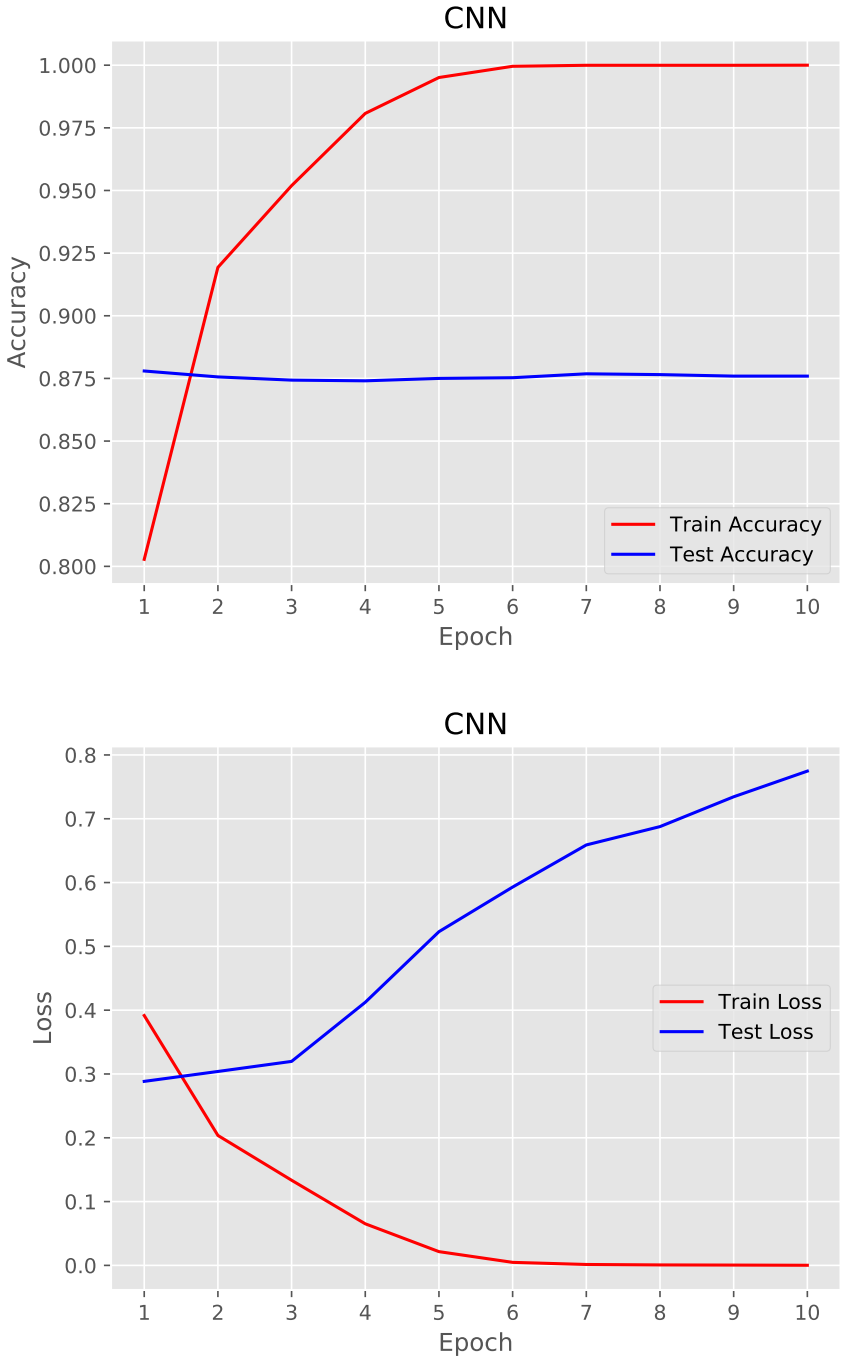

Figure 5: CNN accuracy and loss.

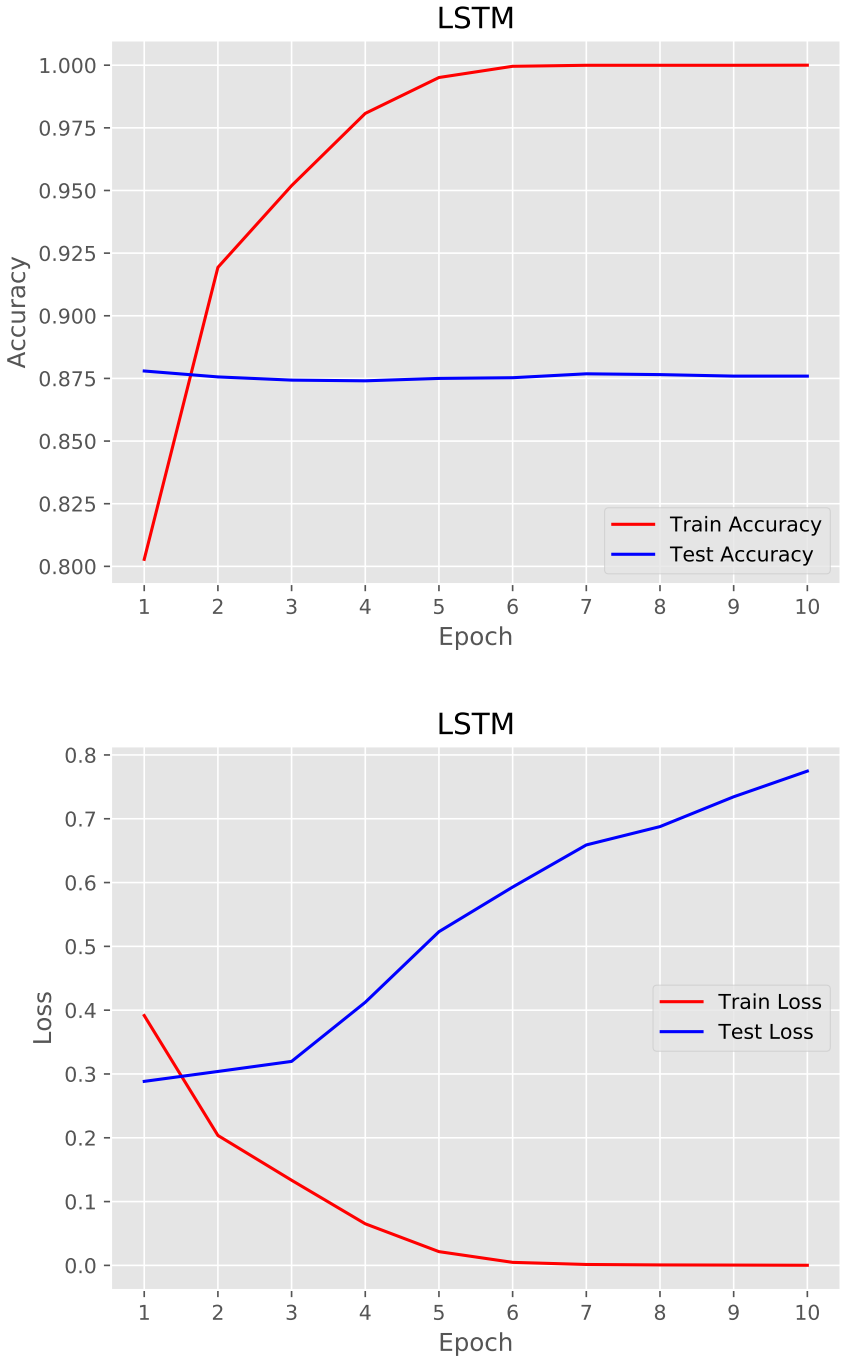

Figure 6: LSTM accuracy and loss.